MIT Media Laboratory, Perceptual Computing Technical Report #242
To appear 2nd CAD-Based Vision Workshop, February 1994, Champion, PA

# Recursive Estimation for CAD Model Recovery

**Ali J. Azarbayejani, Tinsley Galyean, Bradley Horowitz, Alex Pentland**
Perceptual Computing Group & Computer Graphics and Animation Group
The Media Laboratory
Massachusetts Institute of Technology
Room E15-383, 20 Ames Street,
Cambridge MA 02139, U.S.A.

## Abstract

We describe a system for semiautomatically extracting 3-D object models from raw, uncalibrated video. The system utilizes a recursive estimator to accurately recover camera motion, pointwise structure, and camera focal length. Recovered 3-D points are used to compute a piecewise-smooth surface model for the object. Recovered motion and camera geometry are then used along with the original video to texture map the surfaces.

We describe extensions to our previously-reported geometry estimation formulation [2] that incorporate focal length estimation and other improvements, so that accurate estimates of structure and camera motion can be recovered from uncalibrated video cameras. We also discuss the buildup of texture maps from sequences of images, which is important in producing realistic-looking models. Since the computations parallel those used in computer graphics, efficient hardware already exists for doing this part of the modeling.

Examples demonstrate generation of a realistic 3-D texture-mapped model from a video sequence, the post-production manipulation of video, and the combination of computer graphics models with video.

## 1 Introduction

The ability to efficiently extract models from film and video has important implications for computer vision, computer graphics, and digital video. Examples are object matching and recognition, data compression, visualization of camera paths or lighting changes, stereo viewing of video that was originally monocular, and the ability to combine computer graphics and video in a way that correctly handles occlusions, mutual illumination, and other 3-D effects.

Our system recovers textured shape models of rigid objects from image sequences in which the object undergoes some motion relative to the camera. By "textured" we mean that the surfaces of the objects are painted to appear as they do in the original images. The shapes are piecewise smooth, and are computed by fitting parameterized surfaces to a small number of recovered 3-D points on the objects.

Although the shape primitives are low-order, the texture mapping captures many of the visual details required for successful implementation of the above listed applications. For these sorts of tasks, the final product is 2-D images and the 3-D model is only required to be sufficiently accurate to accurately generate novel 2-D views of the object. Because of this, the sort of model generation described here, although simple, has great potential in various fields of visual information processing.

Our system consists of two steps: geometrical estimation and model generation. Geometry estimation yields estimates of the internal camera geometry (focal length), the external scene geometry (pointwise shape), and the relative geometry between the camera and object (translational and rotational motion). Model generation involves computing surfaces using the recovered sparse structure and texture mapping the surfaces using the raw video and the recovered motion and camera geometry.

The first step, geometry estimation, is done using a feature-based approach. Tracking a small number (typically 10 to 20) of features through a motion sequence is sufficient to recover the motion and focal length along with the 3-D locations of the feature points, which (sparsely) represent the shape of the object.

This estimation process is related to a large body of work in computer vision called *structure from motion* (SfM). For arbitrary-length sequences and for real-time processing, recursive methods are clearly desirable, but most of the results in recursive sequence processing have been characterized by either limited applicability or unreliable stability and accuracy performance. Furthermore, most SfM solutions either solve for the orthographic case, which is of limited use for sequences from real cameras, or they require using a calibrated camera, so they would not be useful, for example, in processing existing footage from an unknown camera.

Our formulation eliminates many of the stability and accuracy shortcomings of previous work by considering in detail the representation of the geometry in the imaging process. One important characteristic of our representation, is that it represents both wide-angle and orthographic projection cases and everything in between.

In this paper we extend our previous formulation so that we can estimate focal length in addition to shape and motion. This allows us to recover 3-D models from image sequences from unknown or uncalibrated sources.

The second part of the system, model generation, utilizes the 3-D pointwise structure to generate a surface model. This part requires human input to segment the object into piecewise smooth parts. The recovered camera motion and imaging geometry is used to texture map the surface of the object with the correct visual appearance according to the raw images. The result is a realistic looking model which can be viewed under novel conditions and appears remarkably correct.

The following sections describe the estimator and the model generation process.

## 2 Geometry Estimation

Applications that require real-time or causal processing or require efficiency strongly favor recursive estimation techniques. When results need to be treated statistically it is important to explicitly account for measurement noise and modeling uncertainty. These are both strengths of the extended Kalman filter (EKF) which has been the subject of

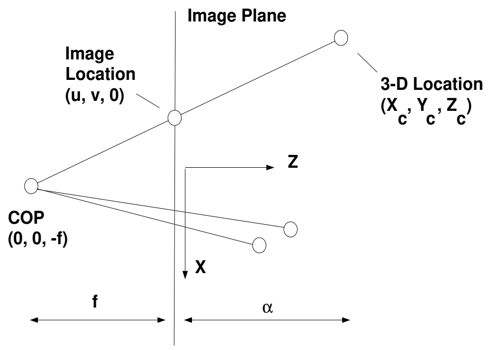


Figure 1: Model of central projection; the center of projection is placed behind, rather than in, the image plane.

much structure and motion estimation research [4; 3; 5; 1; 7; 8; 10; 11; 14].

We use the EKF here, but regardless of the method employed, the fundamental stability and accuracy that can be obtained from a certain set of measurements is highly dependent upon the nature of the measurements and the representation of the data. It is this facet of the problem which we feel is deficient in much previous work. This section discusses five issues of representation related to the geometry of the SfM problem and discusses how our representation contributes to increased stability and accuracy in estimation.

The result is a representation that applies for a camera of any focal length and allows the focal length to be recovered in addition to structure and motion. The representation is minimal in free parameters and is well-conditioned numerically where traditional perspective models break down. The formulation presented here extends our previous work [2] in several significant regards, so that accurate 3-D geometry can be recovered from uncalibrated video sources

## 2.1 Camera Model

Equation 1 models central projection, where $(X_C, Y_C, Z_C)$ is the 3-D location of a point in the camera reference frame, $(u, v)$ is the image location, and $\beta = 1/f$ is the inverse focal length:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} X_C \\ Y_C \end{pmatrix} \frac{1}{1 + Z_C\beta} \qquad (1)$$

Geometrically, this model is identical to the usual model,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} X_C \\ Y_C \end{pmatrix} \frac{f}{Z_C} \qquad (2)$$

with two representational changes. First, as illustrated in Figure 1, the coordinate system origin is fixed at the image plane rather than the center of projection (COP). Second, inverse focal length, $\beta$, is the model parameter rather than focal length.

One important property of this model is that it decouples the representation of the camera from the representation of depth in a desirable way. With the usual model (Equation 2), if the "depth" $Z_C$ of an object is fixed and the "focal length" parameter is altered, *the imaging geometry remains the same and only the scale of the image changes!* (The cone of perspective rays remains fixed while the focal plane translates along the optical axis.) Conversely, note that there is no way to alter the imaging geometry (angles of rays) without also altering the representation of depth ($Z_C$). Thus the parameter $Z_C$ really encodes both the camera imaging geometry *and* depth, while $f$ is really only a scale factor from world units to pixels.

On the other hand, altering the inverse focal length ($\beta$) in Equation 1 alters the imaging geometry independent of the representation of object depth and independent of scale, which is assumed to be unity.[1]

A second important property of this model is that it does not become numerically "ill-conditioned" as focal length becomes large. In fact, orthographic projection, which is only a theoretical mathematical limit with Equation 2, is represented perfectly well with finite parameters ($\beta = 0$) using Equation 1. Thus, all theoretical results that hold for general values of $\beta$ apply uniformly to both perspective and orthographic projection, two cases which are traditionally treated quite differently in computer vision.

Thus, we use this mathematical model of projection with the single parameter $\beta$ to describe the imaging geometry of the camera. If focal length is known, we have a traditional structure from motion problem, with the additional advantage that central and orthographic projection can be treated with the same framework. And if we do not know the focal length *a priori*, estimation of $\beta$ yields an estimate of the imaging geometry (and, of course, works for orthographic as well, in which case $\beta = 0$).

## 2.2 Structure Model

A "feature point" is defined in terms of its image location in the first frame in which it appears. Template matching (of some kind) is used to track the feature based on its appearance, resulting in measurements of image location of the feature in subsequent frames. These measurements contribute to recovery of the 3-D location of the feature point, which has one unknown parameter (outside the intrinsic camera parameters)—that is, the depth in its original camera frame.

The 3-D point can be represented using the Equation 3 below. The first term represents the image location and the second term represents the perspective ray scaled by the unknow depth $\alpha$:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} u\beta \\ v\beta \\ 1 \end{pmatrix} \qquad (3)$$

Pointwise structure, therefore, can be represented with one parameter per point.

This parameterization is considerably lower dimensionality than traditional representations of structure that use three parameters per point [3; 12; 13]. It is critical for estimation stability to use such a minimal parameterization.

To see this, consider the number of degrees of freedom of these structure from motion representations versus the number of available measurements at each frame when $N$ points are tracked. There are six motion parameters and $3N$ structure parameters versus $2N$ constraints from the measurements of new image locations (plus one arbitrary scale constraint). The system is underdetermined for any $N$, with the remarkable and counterintuitive property that the more features (i.e. measurements) that are added, the more underdetermined the system becomes. This leads to an inherently unstable estimation process, regardless of the method employed.

It is important, then, to question the differences in these parameterizations and discover what, if anything, is lost when the more compact representation is used. Although nothing in the above description of structure indicates any lost information, clearly there are fewer degrees of freedom, so there must be a conceptual difference in the way the structure is described. Indeed, the traditional $3N$ parameterization of points arises naturally if one considers a batch of image measurements and makes no assumptions about the causality of feature correspondences. If one assumes only that all the image location measurements for a single feature

[1] Scaling from world units to image units can be implemented with an additional multiplier; but since absolute scale does not affect the basic geometry, scaling is pointless for the internal representation. All lengths can be respresented in pixels and scaled to real lengths later.

come from a single 3-D point, without regard to the measurement process, it is tempting to set up an estimation process to find, for each feature, a 3-D point which might yield the observed measurements with zero-mean error. However, this is not necessarily the best way to construct a cost function for the estimation.

Given the measurement process of *feature tracking*, described above, such a representation is unsuitable, for there is no guarantee that the structure and motion recovered in this way will be consistent with the image-based definition (template) of the feature that was used for tracking. Furthermore, although such a representation may make sense for batch estimations in which the source of measurements is perhaps unknown and there is no order, it may not be as useful for *recursive* estimation, where feature locations are best thought of as measurements of earlier image-based descriptions.

This issue is discussed further in Section 2.6, where it is argued that the $N$-parameter description is exactly right if one is willing to make the assumption that feature tracking measurements have zero-mean error, which is as valid an assumption as any other given no further properties of the feature tracking process. It is further argued that the $3N$-parameter description used in previous formulations is equivalent to our representation, adding the assumption of biased measurements, and trying to estimate the biases.

The basic argument is that if the biases are small, which they typically are in this problem, there is no advantage in destabilizing the estimation process to recover the biases. Thus, we use the $N$-parameter vector

$$\text{(structure)} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix}$$

to represent pointwise structure and we show how biases can be treated in the EKF in the case that large biases are expected.

## 2.3 Translation Model

The translational motion is represented as the 3-D location of the object reference frame relative to the current camera reference frame using the vector

$$\mathbf{t} = \begin{pmatrix} t_X \\ t_Y \\ t_Z \end{pmatrix}.$$

The $t_X$ and $t_Y$ components correspond to directions parallel to the image plane, while the $t_Z$ component corresponds to the depth of the object along the optical axis. As such, the sensitivity of image plane motion to $t_X$ and $t_Y$ motion will be similar, while the sensitivity to $t_Z$ motion will differ, to a level dependent upon the focal length of the imaging geometry.

For typical video camera focal lengths, even with "wide angle" lenses, there is already much less sensitivity to $t_Z$ motion than there is to $(t_X, t_Y)$ motion. For longer focal lengths the sensitivity decreases until in the limiting orthographic case there is zero image plane sensitivity to $t_Z$ motion.

For this reason, $t_Z$ is not represented explicitly in the estimation process. Instead, the product $t_Z\beta$ is estimated. The coordinate frame transformation equation

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C\beta \end{pmatrix} = \begin{pmatrix} t_X \\ t_Y \\ t_Z\beta \end{pmatrix} + \begin{pmatrix} 1 & & \\ & 1 & \\ & & \beta \end{pmatrix} \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (4)$$

combined with Equation 1 demonstrates that only

$$\mathbf{t}^* = \begin{pmatrix} t_X \\ t_Y \\ t_Z\beta \end{pmatrix}$$

is actually required to generate a equation for the image plane measurements $(u, v)$ as a function of the motion, structure, and camera parameters (rotation $\mathbf{R}$ is discussed below).

Furthermore, the sensitivity of $t_Z\beta$ does not degenerate at long focal lengths as does $t_Z$. For example, the sensitivities of the $u$ image coordinate to both $t_Z$ and $t_Z\beta$ are

$$\frac{\partial u}{\partial t_Z} = \frac{-X_C\beta}{(1 + Z_C\beta)^2}$$

and

$$\frac{\partial u}{\partial (t_Z\beta)} = \frac{-X_C}{(1 + Z_C\beta)^2}$$

demonstrating that $t_Z\beta$ remains observable from the measurements and is therefore estimatable for long focal lengths, while $t_Z$ is not ($\beta$ gets small for long focal lengths, zero for orthographic projection).

Thus we parameterize translational motion with the vector $\mathbf{t}^*$. True translation $\mathbf{t}$ can be recovered post-estimation simply by dividing out the focal parameter from $t_Z\beta$, if $\beta$ is non-zero (non-orthographic). The error variance on $t_Z$ will be the error variance on $t_Z\beta$ scaled by $1/\beta^2$, which demonstrates that uncertainty in $t_Z$ will be infinite in the orthographic case and will be large for long focal lengths.

## 2.4 Rotation Model

The 3-D rotation is defined as the relative rotation between the object reference frame and the current camera reference frame. This is represented using a unit quaternion, from which the rotation matrix can be generated:

$$\mathbf{R}(\mathbf{q}) =$$

$$\begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (5)$$

The four elements of the unit quaternion only have three degrees of freedom due to the normality constraint. Thus, all four cannot be estimated independently; only a non-linear constrained minimization will work to recover the quaternion directly. Since the EKF utilizes a linearization at each step, the non-linear normality constraint cannot be rigidly enforced.

However, a 3-parameter incremental rotation representation can be used in the EKF to estimate interframe rotation at each frame. Euler angles centered at zero rotation (or elements of the "rotational velocity vector") are approximately independent and can therefore be used in a system linearization.

The incremental rotation quaternion is a function of these three parameters:

$$\delta\mathbf{q} = \begin{pmatrix} \sqrt{1-\epsilon} \\ \omega_X/2 \\ \omega_Y/2 \\ \omega_Z/2 \end{pmatrix} \quad (6)$$

$$\epsilon = (\omega_X^2 + \omega_Y^2 + \omega_Z^2)/4 \quad (7)$$

This incremental rotation can be computed at each frame and then composed with an external rotation quaternion to maintain an estimate of global rotation. The global quaternion is then used in the linearization process at the next frame.

## 2.5 The Issue of Scale

It is well known that the shape and motion geometry in SfM problems such as this are subject to arbitrary scaling and that this scale factor cannot be recovered. (The imaging geometry $\beta$ *is* recoverable and not subject to this scaling.) In two-frame problems with no information about true lengths in the scene, scale factor is usually set by fixing the length of the "baseline" between the two cameras. This corresponds to the magnitude of the translational motion.

It is equally acceptable to fix any other single length associated with the motion or the structure. In many previous formulations, including [3; 12] some component of the translational motion is fixed at a finite value. This is not a good idea for two reasons. First, if the fixed component, e.g. the magnitude of translation is actually zero, the estimation becomes numerically ill-conditioned. Second, every component of motion is generally dynamic, which means the scale changes at every frame! This is disasterous for stability and also requires some post-process to rectify the scale.

A far better approach to setting the scale is to fix a static parameter. Since we are dealing with rigid objects, all of the shape parameters $\{\alpha_i\}$ are static. Thus, fixing any one of these establishes a uniform scale for all motion and structure parameters over the entire sequence. The result is a well-conditioned, stable representation.

### 2.6 Recursive Estimation

The composite state vector thus consists of $7 + N$ parameters—6 for motion, 1 for camera geometry, and $N$ for structure—where $N$ is the number of features tracked:

$$\mathbf{x} = \begin{pmatrix} t_X \\ t_Y \\ t_Z\beta \\ \omega_X \\ \omega_Y \\ \omega_Z \\ \beta \\ \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} \tag{8}$$

An additional quaternion is required for maintaining a description of the global rotation, as described above. The vector $\mathbf{x}$ is the state vector used in an EKF implementation, where the measurement vector contains the image locations of features in each new frame.

The EKF is considered a common tool now in computer vision and is described in detail in [3; 6; 9], thus will not be discussed here. Instead, we turn attention to the issue of biased noise and how it relates to various representations of object structure.

If one assumes that features are identified in the first frame and that measurements are obtained by comparing new images to the previous images, it is a simple and natural assumption that the measurements might be zero-mean or very close to zero-mean. This thinking leads to the description of structure given in Section 2.2, in which a single unknown for each feature fully describes structure. These parameters can be computed very effectively using the EKF, which assumes zero-mean measurements.

It is common to use Kalman filters even when measurements are not truly zero-mean. Good results can be obtained if the biases are small. However, if the measurements are biased a great deal, results may be inaccurate. In the case of large biases, the biases are observable in the measurements and can therefore be estimated by augmenting the state vector with additional parameters representing the biases of the measurements. In this way, the Kalman filter can theoretically be used to estimate biases in all the measurements!

However, there is a tradeoff between the accuracy that might be gained by estimating bias and the stability of the filter, which is reduced when the state vector is enlarged. When the biases are large, i.e. compared to the standard deviation of the noise, they can be estimated and can contribute to increased accuracy. But if the biases are small compared to the standard deviation they cannot be accurately estimated and they do not affect accuracy much. Thus, it is only worth augmenting the state vector to account for biases when the biases are known to be significant relative to the noise variance.

In our SfM problem, augmenting the state vector to account for bias adds two additional parameters per feature. This results in a geometry representation having a total of $7+3N$ parameters. This is the same number as if one simply used three free spatial parameters for each point, as in [3; 12].

The two interpretations of the problem are clearly related. In the case of biased noise, Equation 3 can be written

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} (1+\alpha\beta)(u+b_u) \\ (1+\alpha\beta)(v+b_v) \\ \alpha \end{pmatrix} \tag{9}$$

where $(b_u, b_v)$ represent the bias. This relation is invertible, i.e.

$$\begin{pmatrix} \alpha \\ b_u \\ b_v \end{pmatrix} = \begin{pmatrix} Z \\ \frac{X}{1+Z\beta} - u \\ \frac{Y}{1+Z\beta} - v \end{pmatrix} \tag{10}$$

so the two representations are equivalent. Thus, our representation is the same as previous representations except that we assume zero-mean-error tracking of features and they assume biased tracking and attempt to estimate the bias.

But, although we can relate our $(\alpha, b_u, b_v)$ parameterization to previous $(X, Y, Z)$ parameterizations, the former is more useful in the EKF estimation framework. It is not clear what priors to enforce in general when structure is represented as three uniform coordinates $(X, Y, Z)$. If a point is represented as $(\alpha, b_u, b_v)$, however, the fact that it is being estimated from an imaging process can be used to infer useful priors on the parameters. Initial error variance on $\alpha$ should be quite high since there is no information, and error variances on $(b_u, b_v)$ should be quite low compared to the pixel size for any imaginable feature tracking algorithm. The parameterization we use, based on $b_u = b_v = 0$, is the result of letting initial error variance on the bias be zero.

## 3 Model Generation

Our models include a geometric description and a description of the visible surface texture. The geometry is modeled using a piecewise smooth surface. The visual appearance is incorporated by texture mapping the surfaces using the original imagery. The following sections describe the computation of object surfaces and texture maps.

### 3.1 Computing the Surfaces

In order to generate a piecewise smooth surface for the object, the original video must be segmented. Segmentation consists of breaking images of the object into regions, each of which can be represented using a single parameterized surface. The feature points in each region are used to compute the surface parameters, using the estimates of 3-D point locations from the geometry EKF. The boundaries of the 2-D region are used to bound the 3-D surface.

Since no general purpose segmentation procedures exist, this step is done manually in our implementation. For each surface to be modeled, a geometric representation must be chosen. In our example, we use polygons for all of the surfaces. For polygons, segmentation consists of identifying the vertices of the polygons in one or more images (each vertex can be selected from a different image, if necessary), which results in 2-D regions corresponding to each surface element. The recovered 3-D feature points are used to estimate a plane for each surface element. The 2-D vertices are inverse projected onto the plane to carve out the 3-D polygon.

Other geometric representations can be treated similarly. Each part of an object can be represented as any parameterized surface (or volume). Segmentation would require identifying the image regions corresponding to each part, allowing the surfaces (or volumes) to be estimated using the recovered 3-D feature points in each region.

### 3.2 Efficient Texture Mapping

The basic concept of recovering the texture map from a given video frame is straightforward. The pixels from each image must be inverse projected onto the appropriate surfaces of the 3-D model. To each surface belongs a texutre map. The texture map locations are computed using the recovered camera motion and focal length from the geometry EKF. Having found the location of a pixel in the texture map, the red, green and blue values from the video pixel can be added to the texture map.

When multiple images of the same surface are available, as in our problem, one would like to use all of the images in the sequence, which means obtaining single texture map values from multiple source images. Described here is an efficient and straightforward way of doing the inverse projection and texture buildup. This implementation works independent of the kind of surface used or the camera parameters.

Building texture maps onto a known structure from image frames is inverse to rendering. Thus a process analagous to rendering can be used to create the texture maps. Because of the similarity to rendering, most of the work that has been done in making rendering efficient also applies here.

The rendering process generates pixels in the image plane by inverse projecting the pixel into the scene and matching the pixel location in the image with a 3-D location on some object. The pixel acquires the color of the object (after shading calculations are done). If the object has texture map on its surface, the correct location is found in the texture map and the value of the pixel is extracted from the texture map instead. This mapping from the object surface location to image map location is typically done by assigning UV coordinate values (between 0 and 1) across the surface of the object, allowing quick lookup of the texture value.

Thus, for each image pixel, rendering establishes a match to an object id and a set of UV coordinates in the object's texture map. Rendering extracts the value from the texture map location and inserts it in the image pixel location. Texture map generation is exactly the same process except the information flow is reversed: the value is taken from the image and inserted in the texture map.

In most cases, the resolution of the texture map is significantly higher than the information provided by a single image, resulting in sparse sampling in the texture map. The texture map can be filled in using various methods, including, for example, bilinear interpolation. By using the topology of the original video frame raster, the samples in the resulting texture map can be thought as corners for warped quadrilaterals. Each quadrilateral has a color at each of it 4 vertices. To fill in the holes in the image map, the quadrilaterals are scan converted, interpolating their vertex colors. This reduces the filtering problem to a scan conversion of 2D quadrilaterals. Since this is also a problem in encounted in computer graphics, this method of filling in texture maps can be done very quickly using graphics hardware.

### 3.3 Filtering Multiple Texture Maps

There are several reasons to use more than one video frame to build the texture maps. First, parts of the scene that are not visible from one point of view may be visible from another viewpoint. Therefore the most complete texture map of a given surface can typically only be built using multiple video frames. Second, a video frame may have a close-up shot of a particular part of the model, in which case there is additional resolution information available for that part of the texture maps. Third, even multiple video frames of the same part of a surface of the same resolution have additional image data for that surface. Therefore, by using multiple video frames it should be possible to extract higher resolution texture maps than would be available from any one video frame.

When multiple frames covering a single object are available, a different texture map can be produced from each video frame, as described above. These texture maps then need to be combined into one texture map that will be the final representation.

The simplest approach is to have the information/pixels from the next texture map overwrite the pixels produced from a previous texture map. This will provide for the highest amount of texture map coverage, meaning that parts seen from earlier frames (but not seen in this frame) will not be distorted. However, with this approach, the texture map data may not be as clear or as high resolution as possible from the available video, and also mis-alignments show up as discontinuities in the texture map.

If there are a large number of original video frames, averaging of the texture maps allows integration of the data, but only if frames are well registered. The average can be a weighted average to account for the differences in resolution between the texture maps. The differences in the UV values from pixel to pixel during the recovery process gives a good metric for the resolution of the samples in the texture map. This information can then be used (on a per pixel basis) to do a weighted average, giving greater weight to the higher-resolution texture maps. One potential problem is that if images are not well registered then averaging tends to blur features. This blurring effect can be minimized by use of the median or mode value for each UV coordinate in the texture map, although this can cause smaller features to disappear.

We have found that our camera motion estimates are sufficiently accurate to provide registration of component texture maps to within a pixel, resulting in a very small amount of blurring using direct averaging. The results in this paper are computed on that basis. It should be noted, however, that the texture maps can probably be improved further by employing these nonlinear image processing algorithms or by performing registration in UV space.

## 4 Results

In this example, models are extracted from a 20-second video clip of a walk-around outside a building (the Media Laboratory, MIT). Figure 3(a) shows several frames of the original digitized video. The visible faces of the building include two outside walls, an overhang, and two lower walls under the overhang.

Over twenty features on the building were tracked. In Figure 3(b) the location of the $(0, 0)$ coordinate of each feature templates is indicated by a small white square. These feature positions in the images are the input to the structure/motion recovery technique described in Section 2. Because of the relatively small window size used in the feature tracking module, feature tracking occurs in near-real-time. However when a feature becomes occluded, or become obscured for some other reason, the match between feature and template becomes poor and manual intervention is sometimes required to correct errors.

Estimation is performed using the EKF described in Section 2, yielding estimates of camera geometry, camera motion, and pointwise structure.

For the three faces of the building with features on them, the recovered 3-D locations of the feature points are used to compute planes in the scene reference frame. The 2-D vertices of the polygons that define each face were manually identified in one video frame and the recovered camera position for that frame is used to back-project the 2-D vertices onto the appropriate 3-D plane, resulting in 3-D vertices for the polygons in scene coordinates. The two remaining polygons were computed from vertices of the existing polygons. The five recovered 3-D polygons are depicted in wireframe in Figure 3(c).

Texture mapping was done as described in Section 3 using multiple frames for each texture as outlined in Section 3.3. For each polygon, twenty-five frames were used to obtain the texture map. Averaging, weighted by the size of the UV projection, was used to combine the texture maps. Figure 3(d) shows the final graphical representation of the building, from several views in the original camera trajectory.

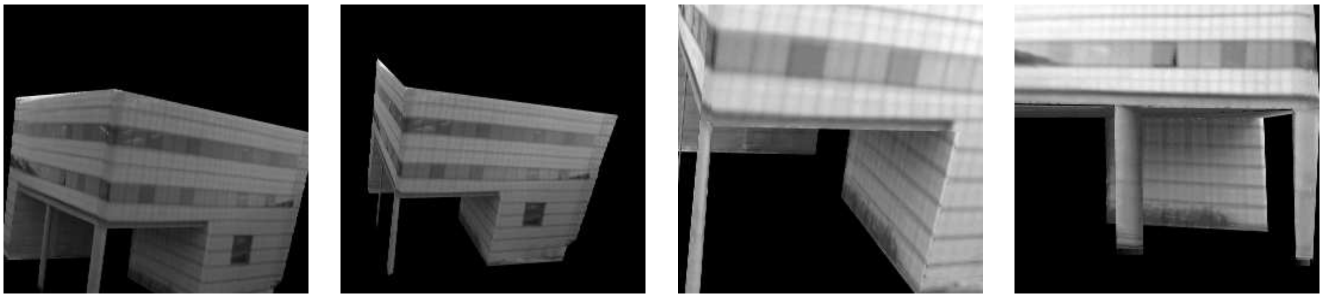

Figure 2: Some novel views from a flyby sequence of the recovered model.

Figure 3(e) shows several frames of an animation, in which the original video of the building is combined with computer graphics. The ability to extract an accurate 3-D model of the scene and the camera position allows occlusions, mutual illumination, and other 3-D effects to be correctly accounted for when combining the 3-D models with the video footage.

Finally, Figure 2 shows several frames of a second animation in which the building is viewed from other camera angles, demonstrating the capability to view the object under novel conditions.

## 5 Conclusion

We have presented a system for recovering models representing the 3-D geometry and visual appearance of objects from image sequences. We believe this work makes significant contributions to the field on several fronts. The estimation process is reliably stable and accurate, due to careful consideration of the geometry of the problem. The estimation recovers focal length and is well-conditioned for any focal length, so virtually any image sequence from any camera can be processed. The texture mapping procedure is designed using standard rendering techniques, so that powerful computer graphics techniques and hardware can be used directly.

All parts of the system are completely automated except for feature tracking and image segmentation. It is very difficult to automate these two processes in a robust fashion, but the un-automated tasks are trivial for a human. Thus our tools are currently configured to be an interactive application, but can directly incorporate future algorithms that can reliably perform feature tracking and segmentation.

Real-time systems could eventually be used for mobile robots and on-line video compression, but even interactive systems are important because they provide an efficient way to creating realistic appearing models for object recognition, computer graphics and image compression. Our examples illustrate that the resulting models are indeed visually convincing. Consequently realistic re-rendering for robotics, graphics, and video compression applications is now closer to being a practical, economically-viable reality.

## References


[1] Nicholas Ayache and Olivier Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. Robotics Automation*, 5(6):804–819, 1989.

[2] Ali J. Azarbayejani, Bradley Horowitz, and Alex Pentland. Recursive estimation of structure and motion using relative orientation constraints. In *1993 IEEE Conference on Computer Vision and Pattern Recognition*, New York, 1993. IEEE Computer Society.

[3] Ted J. Broida, S. Chandrashekhar, and Rama Chellappa. Recursive estimation of 3-d motion from a monocular image sequence. *IEEE Trans. Aerosp. Electron. Syst.*, 26(4):639–656, July 1990.

[4] Ted J. Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[5] Ted J. Broida and Rama Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):497–513, June 1991.

[6] Robert Grover Brown. *Introduction to Random Signal Analysis and Kalman Filtering.* John Wiley & Sons, New York, 1983.

[7] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240, 1988.

[8] Olivier D. Faugeras, Nicholas Ayache, and B. Faverjon. Building visual maps by combining noisy stereo measurements. In *Proc. IEEE Conf. on Robotics and Automation*, April 1986. (San Francisco, CA.).

[9] Arthur Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[10] Joachim Heel. Temporally integrated surface reconstruction. In *ICCV '90.* IEEE, 1990.

[11] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–236, 1989.

[12] J. Oliensis and J. Inigo Thomas. Incorporating motion error in multi-frame structure from motion. In *IEEE Workshop on Visual Motion*, pages 8–13, Los Alamitos, CA, October 1991. IEEE Computer Society, IEEE Computer Society Press. (Nassau Inn, Princeton, NJ.).

[13] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Optimal motion and structure estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 144–152, June 1989. (San Diego, CA.).

[14] G-S. Young and Rama Chellappa. 3-d motion estimation using a sequence of noisy stereo images: Models, estimation and uniqueness. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):735–759, January 1990.

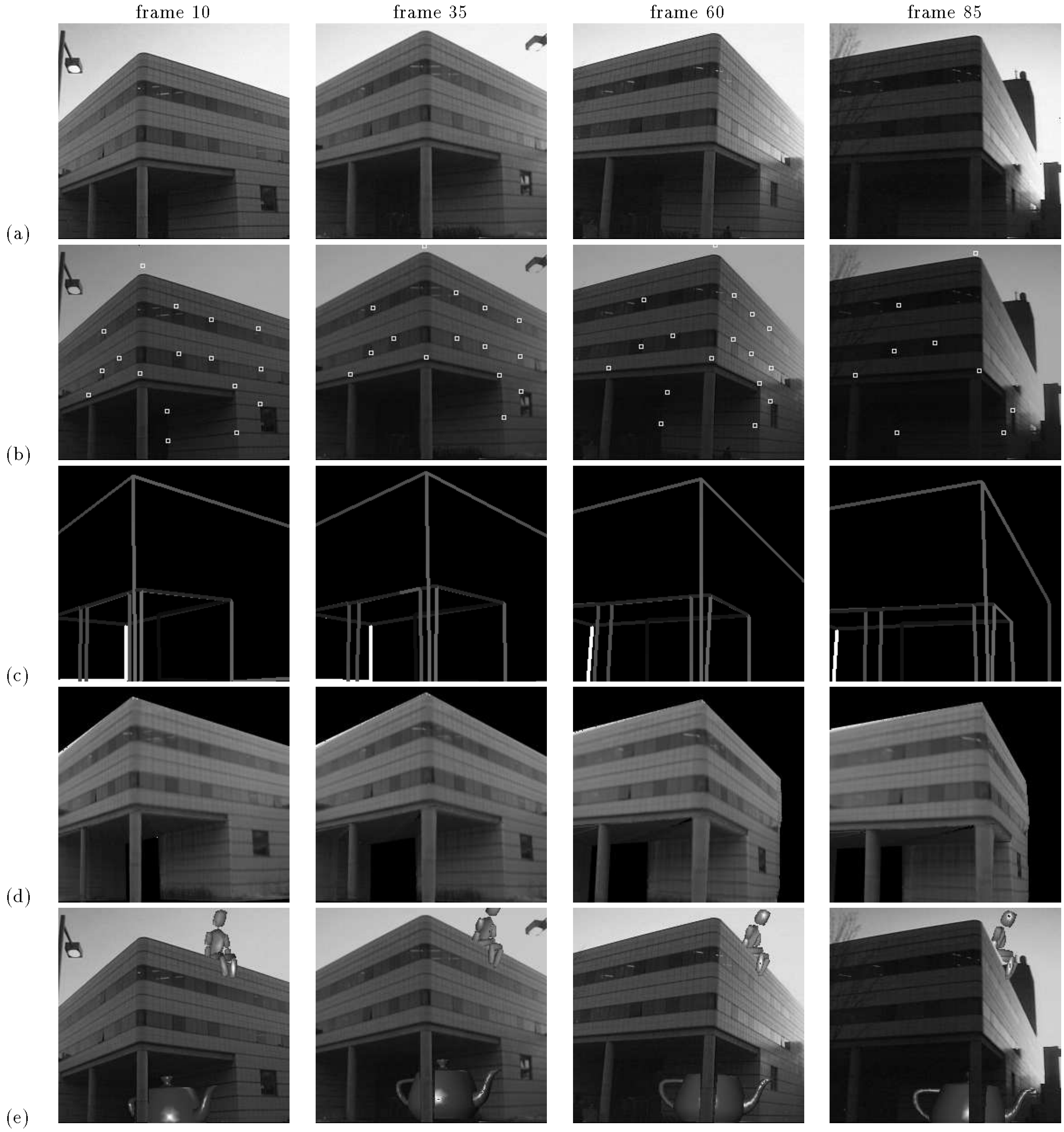


Figure 3: (a) Original video. (b) The features are tracked in the video sequence using normalized correlation. Features are added and deleted as necessary. (c) 3-D polygons are obtained by segmenting a 2-D image and back-projecting the vertices onto a 3-D plane. The plane for each polygon is computed from the recovered 3-D points corresponding to image features in the 2-D polygon. (d) Texture maps are obtained by projecting the video onto the 3-D polygons to obtain the final model. Note that since multiple frames are used to obtain the texture maps, the lamp post passing in front of the building between frames 10 and 35 is filtered out of texture map. (e) Putting graphics into video. The “teapot” and the “dude” are rendered with the model, then separated from the model and composited with the original video.